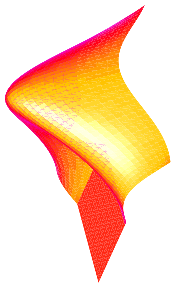
MATLAB® Toolbox

MIMO

KAMIL ANIS

Rev. 2.2.0

Department of Radioelectronics
Faculty of Electrical Engineering
Czech Tech. University in Prague
http://radio.feld.cvut.cz/



Computation

Visualization

Programming

Typographic Conventions

`GLOBAL`	Global variable. Face: black, monospaced, uppercase.
`local`	Regular variable. Face: black, monospaced.
`function`	Reference to the MIMO toolbox function. Face: blue, monospaced, underlined.
`function`	Reference to the regular MATLAB function. This is equivalent to your local HTML MATLAB documentation. By clicking on this reference a web browser will be invoked. Face: blue, monospaced.
Property	Property name. Face: bold.
[2]	Link to the bibliographic reference. Face: red.
Tab. 1	Regular link. Face: blue.

When a list of possible values in form `{PropertyValue1} | PropertyValue2 | PropertyValue3` is available the default value is surrounded by the curly braces. Not always there must be a default value there.

General Properties

The System Model

Principal scheme of the MIMO toolbox is shown in Fig. 1. Desired pattern taken from a discrete alphabet is used to generate testing data. The data are then encoded in the channel encoder and split into the multiple parallel streams. The signal at the modulator's output is a complex evelope of real bandpass signal sampled at certain sampling frequency. received signal is corrupted by the fading in the MIMO channel. The signal than enters matched filter and is downsampled once per a symbol period. This resampled signal (sufficient statistics) is than passed along to the detector in order to evaluate the data estimations. Both the encoder and the detector operate over the look-up table so once the encoding scheme at the transmitter gets changed the receiver operation mode gets changed as well.[1]

Note The whole system is tailored to operate in the discrete time domain (Dt) but all the quantities behave as it would be continous time (Ct) system.

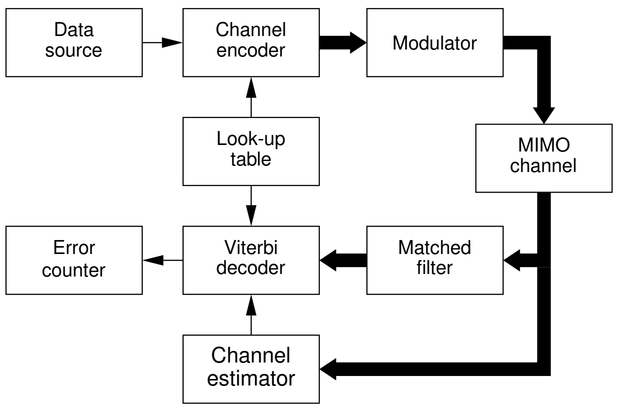


Fig. 1: Block scheme of MIMO toolbox

Running the Modules

All the functions match the MATLAB's conventions. Thus the usage is as simple as in the case of regular MATLAB's functions.

```
[y1,y2,...] = function(x1,x2,...)
```

Some functions can have additional properties that control function behavior. The notation must always take the form of pairs i.e.

```
y = function(x,'PropertyName',PropertyValue,...)
```

No matter what is a pair order. So once you memorize function properties you can combine them at your disposal as long as the *'PropertyName'* has appropriate `PropertyValue` assigned.

The *'PropertyName'* is assumed to be a string compounded from one or more words with first capitalized letter—no spaces between words are allowed. `PropertyValue` can be either number or string. For instance of string the lower case letters are usually used.

[1] Channel estimator has been temporarily removed.

General Properties

Note Some function properties are allowed to be controlled globally in order to provide so called *system-wide settings* such as function echoes etc. Typicaly the value of `SMPLFREQ` can be set globally since it is assumed that all the system devices operate on the common sampling frequency. The global setting may be overrided by the local setting.

In fact all the modules are connected together via one (main) script called `mimo`. This let's you better focus your attention on the particular block development. A unified data exchange format is used throughout all toolbox devices.

Note Many toolbox functions produce internal echoes during the computation. The **Echo** property is turned `off` by default. So if you have any doubts about function behaviour, turn the **Echo** property on. Sometimes it's really useful in order to trace for the bugs.

Data Exchange Format

General structure of the MIMO toolbox data exchange format is shown in Fig. 2. Here the benefit of multi-dimensional arrays is exploited. The advantage is that a frame synchronizer is not needed—all the devices always know when the frame begins and ends. In the other hand large matrices are not the best solution to measure a code performance over hundred thousands of symbols. The idea is to design and verify code in MATLAB environment and measure its performace under more suitable system such as Cadence SPW. Hope that figure tells all and no more explanation is needed.

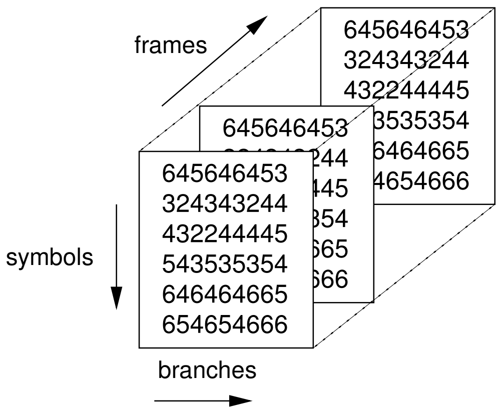


Fig. 2: MIMO toolbox data exchange format

Context Help

All the functions are equipped with a context help. You just type

```
help ⟨function⟩
```

right into the MATLAB's command window. A more detailed description should be found in the Function Reference section.

MIMO Toolbox Function Reference

brmet

User defined branch metric

Syntax

```
rho = bremet(r,q,alpha)
```

Description

`rho = bremet(r,q,alpha)` evaluates branch metric and returns the result as a column vector for all testing channel symbols `q`. The entries of `rho` variable correspond to the squared Euclidean distance of received signal `r` and testing channel symbols `q`. The value of the `rho` is by default computed from the following formula accordingly to [2]

$$\rho = \sum_{j=1}^{m} \left| r_t^j - \sum_{i=1}^{n} \alpha_{i,j} q_t^i \right|^2$$

where m is the number of receive antennas, n is the number of transmit antennas, r_t^j is the sample of the signal at time t at j-th receive antenna `r`, $\alpha_{i,j}$ is complex path fading from i-th transmit antenna to the j-th receive antenna (`alpha`) and q_t^i is a testing channel symbol `q`.

See Also

`detect`, `channel`.

← channel →

channel

MIMO channel model

Syntax

```
y = channel(x)
[y,alpha] = channel(x)
[y,alpha] = channel(x,'PropertyName',PropertyValue,...)
```

Description

`y = channel(x)` corrupts the input signal `x` by the channel fading and mixtures the signal accordingly to the Fig. 3.

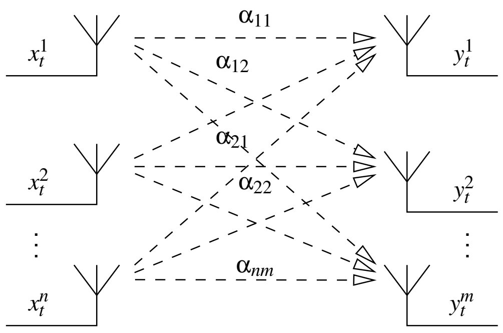


Fig. 3: MIMO channel model

`[y,alpha] = channel(x)` same as above but returns the matrix of complex path fading. This matrix is needed by the detector.

Channel Property Descriptions

Arrangement `[n m]`

MIMO channel arrangement. Tells the channel that there are `n` transmit and `m` receive antennas. Default setting for the Tarokh's et al. space-time codes is 2×2.

Fading `{'none'} | 'awgn' | 'rayleigh'`

Channel fading type. Fading is assumed to be quasistatic and flat i.e. the channel vary from one frame to another. A signal-to-noise ratio in decibels must be set unlike a `'none'` fading is chosen.

Note When the `'none'` option is chosen than there are no noise in the channel, however the direct signals will be combined at the receiver input (see Fig. 3).

SignalPower `Ps`

Modulated signal power. The value of the signal power is necessary to determine correct noise level accordingly to the desired signal-to-noise ratio.

SNR `snr`

Signal-to-noise ratio. Sets the signal-to-noise ratio in [dB]. The `snr` over the MIMO channel is to be defined

$$\mathrm{SNR}_{\mathrm{MIMO}} = N_T \frac{E_{\mathrm{b}}}{N_0}$$

where N_T is the number of transmit antennas, E_{b} is the average energy per one bit and $N_0/2$ is the white noise power spectral density.

SymbolTime `Ts`

Symbol time duration. This value is needed in order to determine average energy per a symbol E_{sr}.

SamplingPeriod `Tp`

System sampling period. Since the system is tailored to operate in discrete time domain this value is used to determine a noise variance such as it would operate in the continuous time domain.

Echo `'on' | {'off'}`

Function echo. Toggles internal function echo.

See Also

detect, est.

← brmet count →

count

Error counter

Syntax

```
err = count(d,d_e)
[err,ser] = count(d,d_e)
[err,ser] = count(d,d_e,'PropertyName',PropertyValue,...)
```

Description

`err = count(d,d_e)` simply counts an error occurrence of data estimations `d_e`.

`[err,ser] = count(d,d_e)` same as above, but also gives a symbol error rate in decibels.

Count Property Descriptions

Echo	`'on'	{'off'}`

Function echo. Displays the results of the error counting process.

```
COUNT: Total errors counted -> ?.
       Symbol error rate -> ? [dB].
       System reliability -> ? %.
```

SERLowerBound	`{1e-6}`

Symbol error rate lower bound. This value is substitued instead of $-\infty$ when no errors counted, so that the measured symbol error rate can be plotted into the figure. Note that

$$\mathrm{SER}_{\mathrm{dB}} = 10\log(\mathrm{SER}).$$

If needed `'SERLowerBound'` can be set to `-Inf`.

See Also

`detect`, `source`.

← channel detect →

detect

Multidimensional data detector

Syntax

```
d_e = detect(s,dlt,slt,alpha)
[d_e,s_e] = detect(...)
[d_e,s_e] = detect(...,'PropertyName',PropertyValue,...)
```

Description

`d_e = detect(s,dlt,slt,alpha)` performs the maximum likelihood sequence estimation (MLSE) i.e. Viterbi algorithm on the received signal and returns the data estimations. The look-up tables `dlt` and `slt` are used together with the external function `brmet` which is called during the computation to evaluate branch metric. `alpha` includes a channel complex path fadings.

`[d_e,s_e] = detect(...)` also returns a matrix including the track with most probable path in the code trellis. This matrix is required when the decoding process suppose to be displayed with `disptrell` function.

Echo	`'on'	{'off'}`

Function echo. Toggles internal function echo. When the echo is enabled than the following summary is displayed.

```
DETECT: Performing data detection.
        This may take a while. Please wait...
        Total decoding complexity -> ? steps.
        Total elapsed time -> ? hrs, ? min, ? sec.
```

See Also

`brmet`, `disptrell`, `mfilter`.

← count dispdes →

dispdes

Display designed code in a graphical form

Syntax

```
    dispdes(dlt,slt)
```

Description

`dispdes(dlt,slt)` picks-up the entries from `dlt` and `slt` look-up tables and displays designed code in a graphical form. The figure may be than exported into various graphical formats including JPEG, GIF, PostScript® and inserted into the report. An example for 2 space-time code, 4PSK, 8-states 2 bit/s/Hz is shown in the Fig. 4.

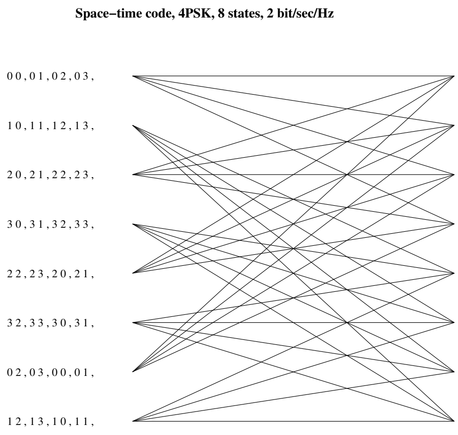


Fig. 4: Code graphical view obtained form `dispdes` function

The left hand side labels (pairs) correspond to the channel symbols to be transmitted over the first and second antenna respectively. The pair order is given by the order of the incoming data symbol

$$\underbrace{\{0,1,2,3\}}_{\text{data}} \mapsto \underbrace{\{1,2,3,4\}}_{\text{position}}.$$

Of course the transitions between the current state and the next state form the code trellis.

See Also

`ltable`.

← detect disptrell →

disptrell

Display decoding process in a graphical form

Syntax

```
disptrell(dlt,slt,d_e,s_e)
disptrell(...,'PropertyName',PropertyValue,...)
```

Description

`disptrell(dlt,slt,d_e,s_e)` displays the most probable path and data estimations in decoding process. `dlt` and `slt` look-up tables are required to display code trellis with all possible transitions. The `disptrell` function doesn't affect the detection process. It's just a tool intended for lectures and presentations. As it belongs to the *visual layer* it's not included in the `detect` function in order to speed up the decoding process.

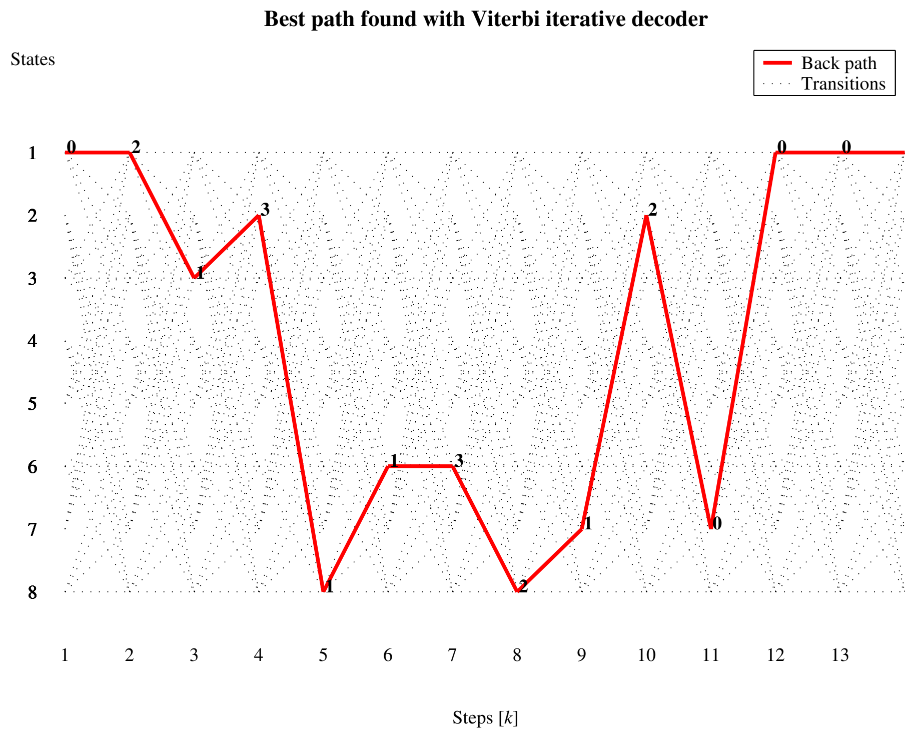


Fig. 5: Decoding process and the most probable path obtained from the `disptrell` function

Note Only one frame per a figure can be displayed! When the frame length exceeds the value of internal variable `FRAMECUT` the rest of the frame is than ommited. Default value for the `FRAMECUT` is 15 symbols (steps). This value is recommended and user is not adviced to change it.

Disptrell Property Descriptions

Shrink $0 < \texttt{SHRINK} < \{0.5\}$

Figure fine tuning. When this option is chosen than a figure gets shrink. Default value for `SHRINK` is 0.5. This corresponds to the 100 % of the figure's height. An example of `disptrell` function is shown in Fig. 5.

Echo `'on' | {'off'}`

Function echo. Toggles internal function echo.

See Also

`detect`.

← dispdes fashion →

fashion

MIMO toolbox fashion file

Syntax

```
fashion
```

Description

`fashion` includes various (recommended) graphics settings for the MIMO toolbox such as colors, line styles and fonts settings. All the objects are set on the root level. This allows you to have a unified fashion-look of the *visual layer* and there's no need to set these properties for each figure or object generated by the `mimo` toolbox. If you want to have these settings to be the MATLAB's defaults, copy its content into the file called `startup.m`. On the UNIX platforms this file should be located in `$HOME/matlab/` directory.

← `disptrell` `hms` →

hms

Convert time to hour-min-sec format

Syntax

```
[str] = hms(time)
[h,m,s] = hms(time)
[...] = hms(time,'PropertyName',PropertyValue,...)
```

Description

`[str] = hms(time)` takes the `time` variable in seconds and returns the string of the form `'xx hours, xx min, xx sec'`. Note that `'Format'` property must be set to `'string'`.

`[h,m,s] = hms(time)` simlply returns hour-min-sec time format.

Hms Property List

Format `{'numeric'} | 'string'`

Output format. Determines whether output will be returned in numeric or string format.

See Also

`tic`, `toc`, `clock`.

← fashion iecho →

iecho

Indent echo

Syntax

```
[idt,tag] = iecho(name)
```

Description

`[idt,tag] = iecho(name)` is a useful utility which allows function echoes to be indented. The script returns an empty string `idt` such that is inserted in front of displayed text. Than you don't need to take care about prompt indenting any more. To change indent spacing set the `INDENT` variable to the desired value. Default value for `INDENT` is 12 (characters). This means that 12 blank spaces will inserted in front of the `name`. `INDENT` may also be set globally.

Example

```
name = 'FUNCTION';
[idt,tag] = iecho(name);
disp(' ');
disp([tag,'Some echo on the 1st line.']);
disp([idt,'Some echo on the 2nd line.']);
disp([idt,'Some echo on the 3rd line.']);
disp([idt,'Some echo on the 4th line.']);
disp(' ');
```

produces

```
FUNCTION: Some echo on the 1st line.
          Some echo on the 2nd line.
          Some echo on the 3rd line.
          Some echo on the 4th line.
```

← hms logreport →

logreport

Create simulation log-file

Syntax

```
logreport
```

Description

`logreport` simply archives simulation environment variables and running conditions into a log-file named `mimotools.log`. In fact `logreport` script is a part of the main `mimo` script and therefore it must not work when running alone.

See also `mimo`, `mesq`, `post`.

← iecho ltable →

ltable

Space-time code look-up table generator

Syntax

```
[dlt,slt] = latble(md,s)
[dlt,slt] = ltable(md,s,'PropertyName',PropertyValue,...)
```

Description

`[dlt,slt] = latble(md,s)` creates `md`, `s`-states space-time code data look-up table `dlt` and state look-up table `slt`. `md` corresponds to the number of the constellation signals e.g. `md`-PSK and `md`-QAM. The summary of available designs is listed in Tab. 1.

Design No.	MD	Modulation	States	Method
1.	4	PSK	4	'att'
2.	4	PSK	8	'att'
3.	4	PSK	16	'att'
4.	4	PSK	32	'att'
5.	8	PSK	8	'att'
6.	8	PSK	8	'delay'
7.	8	PSK	16	'att'
8.	8	PSK	32	'att'
9.	16	QAM	16	'att'
10.	16	QAM	16	'delay'
11.	16	QAM	16	'ext'

Tab. 1: Designs available via `ltable` function

Generally `dlt` look-up table is created as a three dimensional array where the rows correspond to the code states, and columns to the channel symbols to be transmitted over appropriate MIMO branch (antenna). Imagine Fig. 2 with the following substitutions: symbols $\mapsto$ states; branches $\mapsto$ channel symbols; frames $\mapsto$ branches i.e. space dimensions. For a multidimensional schemes this table must be provided to the channel encoder as many times as is the number of the MIMO system branches. Hence the third array dimension is used to allow a distinct channel symbol assignment to the corresponding branch. Trellis transitions are defined via `slt` table which is simply filled with entries that correspond to the following state. Thus the channel encoder simply determines the position based on the current state (row) and the incoming data (column). This value is stored to be the next encoder state.

Example

We design AT&T 2 space-time code, 4PSK, 8-states, 2 bit/sec/Hz. In such a case there are two branches (antennas) at the transmitter. So `slt` will contain two encoding tables $\texttt{dlt}^1$ and $\texttt{dlt}^2$. Appropriate look-up tables look like this

$$\texttt{dlt}^1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}; \quad \texttt{dlt}^2 = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \\ 2 & 3 & 0 & 1 \\ 2 & 3 & 0 & 1 \\ 2 & 3 & 0 & 1 \\ 2 & 3 & 0 & 1 \end{bmatrix}; \quad \texttt{slt} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix}$$

Ltable Property Descriptions

Method `{'att'} | 'delay' | 'ext'`

Design method. When the option `'att'` is chosen than Tarokh's et. al AT&T space-time code is designed [2]. Option `'delay'` forces the `ltable` to design delay diversity code proposed by Wittneben [3]. Moreover `ltable` can operate over user defined pattern defined via external file. So when the option `'ext'` is chosen `ltable` unwraps the pattern and make a code. Usually the pattern is taken from a block code (see [2, page 759]). You type `type stc_bc16.txt` right to the command window to see how the entries in the extern file are stored.

Echo `'on' | {'off'}`

Function echo. Toggles internal function echo.

See Also

`dispdes`, `space`.

← logreport makepulse →

makepulse

Modulation impulse design

Syntax

```
h = makepulse('PropertyName',PropertyValue,...)
```

Description

`h = makepulse('PropertyName',PropertyValue,...)` returns the impulse response of desired modulation impulse.

Makepulse Property Descriptions

Shortening `L`

Modulation impulse shortening. `L` corresponds to the impulse shortening in sense of $\mathtt{L}\cdot\mathtt{Ns}$.

SymbolTime `Ts`

A symbol time duration. `TIME` stands for a symbol time duration in seconds. For example if $\mathtt{L}=8$ than makepulse gives the impulse of total length equal to $8\cdot\mathtt{Ns}+1$ (see Fig. 6). One sample is added to ensure symbol symmetry and correct sampling.

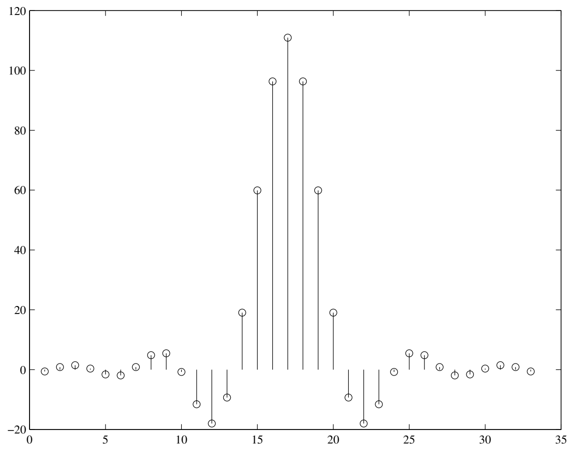


Fig. 6: Samples of square-root raised cosine impulse obtained from makepulse function; $\mathtt{L}=8$, $\mathtt{Ns}=4$, total impulse length$=4\cdot8+1=33$ samples, $\mathtt{alpha}=0.4$, $\mathtt{Ts}=10^{-4}$

Samples `Ns`

Samples per a symbol. `SAMPLES` corresponds to the number of the samples per one symbol.

Shape `{'rrc'} | 'rect'`

Modulation impulse shape. The options `'rrc'` and `'rect'` stand for a root raised cosine and rectangular modulation impulse respectively. When a `'rrc'` option is chosen than a `alpha` factor must be specified. All the impulses are designed to have a unit energy. Root raised cosine impulse is to be defined [1]

$$h(t) = \frac{1}{\sqrt{Ts}\left(1 - 16\alpha^2 \frac{t^2}{T_s^2}\right)} \left(\frac{\sin\left((1-\alpha)\frac{\pi t}{T_s}\right)}{\frac{\pi t}{T_s}} + \frac{4\alpha \cos\left((1+\alpha)\frac{\pi t}{T_s}\right)}{\pi} \right)$$

where T_s is a symbol time duration, α is the roll-off factor; $0 < \alpha < 1$.

Rectangular impulse (REC1) is to be defined

$$h(t) = \frac{1}{\sqrt{T_s}}; \qquad t \in \langle 0, T_s)$$

RollOff $0 < \texttt{alpha} < 1.$

Roll-off factor. Sets the roll-off factor of the root raised cosine modulation impulse.

Echo `'on' | {'off'}`

Function echo. Toggles internal function echo.

See Also

`modul`.

← ltable mesg →

mesg

Display progress information

Syntax

```
mesg
```

Description

`mesg` subroutine displays progress status when performing MIMO system performance measurement. Therefore it makes no sense to run this routine alone.

See Also

mimo, progress.

← makepulse mfilter →

mfilter

Matched filter

Syntax

```
    y = mfilter(x,'PropertyName',PropertyValue,...)
```

Description

`y = mfilter(x)` passes the incoming signal `x` along the matched filter and resamples filtered signal once per a symbol period.

Mfilter Property Descriptions

SymbolSamples `Ns`

Samples per a symbol. Determines the number of samples per a symbol.

SamplePeriod `Tp`

System sampling frequency. Sets the system sampling frequency.

Gain `G`

Constellation expansion factor. Corresponds to the constellation expansion factor computed by the `rescale` function.

Pulse `H`

Modulation impulse impulse response. The samples of modulation impulse are used to be a matched filter impulse response.

See Also

`makepulse`, `rescale`, `detect`.

← mesg mimo →

mimo

Run MIMO system simulation

Syntax

```
mimo
```

Description

mimo is the main simulation script where all the blocks are glued together. Refer to the Fig. 1. This is the body of the program which allows you to set various simulation conditions and change wide range of parameters. Modular architecture let's you easy focus your effort on the particular block development while the rest of the system is not affected by this module. Other modules can be integrated into the body very easily. You just modify the simulation parameters in the script header. mimo script is also suited for a code performance measurement. In such a case you should disable function ECHOes and enable PROG_INFO instead. This keeps you informed about the simulation run, which usually takes a long time. You may also do some post-processing by calling a post function or archive the simulation report into a log-file.

See Also

post, mesg, logreport and the script body to learn more.

← mfilter modul →

modul

Multidimensional digital modulator

Syntax

```
    s = modul(q,md,'PropertyName',PropertyValue,...)
```

Description

`s = modul(q,md)` performs linear memoryless digital modulation of channel symbols `q`. The output signal sampled `Ns`-times per symbol period is assumed to be a complex envelope of the real bandpass signal. `md` corresponds to the number of constellation signals. Possible values for `md` are `4|8|16` which corresponds to 4PSK, 8PSK and 16QAM constellations respectively. The ordering of 16QAM constellation signals is defined via external file `qam16.txt` and may be arbitrarily modified.

Modul Property Descriptions

Samples `Ns`

Samples per a symbol. Determines the number of samples per a symbol.

SamplePeriod `Tp`

System sampling frequency. Sets the system sampling frequency.

Gain `G`

Constellation expansion factor. Corresponds to the constellation expansion factor computed by the `rescale` function.

Pulse `H`

Modulation impulse impulse response. The samples of modulation impulse are used to be a matched filter impulse response.

Echo `'on' | {'off'}`

Function echo. Toggles internal function echo.

See Also

`makepulse`, `rescale`.

← mimo post →

post

Data post-processing

Syntax

```
post
```

Description

`post` simply plots the system performance so that it can be directly exported into the file.

See Also

`mimo`, `logreport`.

← mimo progress →

progress

Create a progress bar

Syntax

```
    bs = progress(percent)
    bs = progress(percent,'PropertyName',PropertyValue,...)
```

Description

`bs = progress(percent)` returns a string showing the ratio of done/remaining jobs.

Example

```
    bs = progress(22.2);

    bs =
         [|||||     22.2 %          ]
```

Progress Property Descriptions

BarSize `{48}`

Progress bar size. Number of chracters a progress bar will be created with. Default value is 48.

BodyChar `{'|'}`

Progress bar body character. Allows to assign a user defined character the body of the progress bar will be drawn with. For example you might use `'='` instead of default `'|'`.

See Also

`makepulse`, `rescale`.

← post rescale →

rescale

Constellation expansion factor

Syntax

```
g = rescale(md,Es)
g = rescale(md,Es,'PropertyName',PropertyValue,...)
```

Description

`g = rescale(Es)` simply computes an expansion constellation factor sets the modulated signal power. Symbol energy `Es` is assumed to be equal `Ts`·`Ps` i.e. the symbol time duration and transmitted signal power respectively. `md` corresponds to the number of constellation signals. Possible values for `md` are `4|8|16` which corresponds to 4PSK, 8PSK and 16QAM constellations respectively. There are two different formulas in the `rescale` function [1].

$$g_{\mathrm{PSK}} = \sqrt{2\mathrm{Es}}$$

$$g_{\mathrm{QAM}} = \sqrt{\frac{6\mathrm{Es}}{\mathtt{md}_{\mathrm{I}}^2 + \mathtt{md}_{\mathrm{Q}}^2 - 2}}$$

See Also

`modul`.

← progress source →

source

Generate data source

Syntax

```
d = source(k,f,md,zf)
d = source(...,'PropertyName',PropertyValue,...)
```

Description

`d = source(k,f,md,zf)` creates a data source consists of desired number of `f` frames each having `k` symbols. `md` determines the range of integers (data) the source will be filled with. For instance of $\mathtt{md} = 4$, `d` will contain numbers within a set $\{0,1,2,3\}$. `zf` corresponds to the number of zeros that will be appended at the end of each frame. This ensures the encoder will be forced into the zero state at the beginning and end of each frame.

Source Property Descriptions

Pattern	`{'rand'}	'ramp'	'const'`

Data source pattern. This property lets you specify a desired data pattern in order to perform various system investigations. Note that `'rand'` option produces uniformly distributed data symbols.

Const	$\mathtt{c} \in \{0,\ldots,\mathtt{md}-1\}$

Constant value. This value is required when the `'const'` pattern is chosen. This corresponds to the frame with constant codeword.

Echo	`'on'	{'off'}`

Function echo. Toggles internal function echo.

See Also

`space`.

← rescale space →

space

Space-time channel encoder

Syntax

```
q = space(d,dlt,slt)
q = space(...,'PropertyName',PropertyValue,...)
```

Description

`q = space(d,dlt,slt)` performs the data (`d`) encoding based on the `dlt` and `slt` look-up tables. The tables for AT&T space-time codes these tables may be easily obtained from the `ltable` function. In fact the structure of look-up tables is as general as possible. By the way the encoder doesn't care about the content of the look-up tables. This feature allows you to force the encoder's operation mode into the arbitrary scheme. Even an encoding based on block codes may be performed using the `space` function. This is when the `slt` table is filled with equal entries. There is also a zero-force check performed during each frame encoding process.

Space Property Descriptions

Display `{'off'} | 'report'`

Display encoding report. When the option `'report'` is enabled than `space` returns the full encoding report in the form of table. This feature is available only when the internal function echo is enabled.

Echo `'on' | {'off'}`

Function echo. Toggles internal function echo.

See Also

`ltable`, `dispdes`.

← source →

References

[1] SÝKORA, J. *Digital Radio Communication II*, 1.1.0 ed. Czech Technical University Press, March 1998.

[2] TAROKH, V., SESHADRI, N., AND CALDEBRANK, A. R. Space-time codes for high data rate wireless communication. *IEEE Transactions On Information Theory 44*, 2 (March 1998).

[3] WITTNEBEN, A. Base station modulation diversity for digital simulcast. In *VTC* (May 1993), IEEE.